Vending Machine System

DELIVERABLE 1

By Zeel Gajjar

Scenario

The project simulates a real-life vending machine. A user approaches the vending machine, inserts cash, selects a product (such as a drink or snack), and receives the item if sufficient funds are available. The vending machine will return any remaining change. Operators can restock items, update product prices, and review profit sheets. The system will support various users and manage product inventory, cash handling, and simple data recording.

Design Paradigm:

- Handle cash input: insert money, return change, and cancel with refund
- Allow selection and purchase of food items (drinks/snacks)
- Dispense items and display success or error messages (e.g., insufficient funds)
- Display available products with names, prices, and categories
- Compare and sort items (by price or name)
- Update inventory after purchases or restocking
- Enable operators to restock items and update product prices
- Record transactions and calculate total profits (with optional file output)

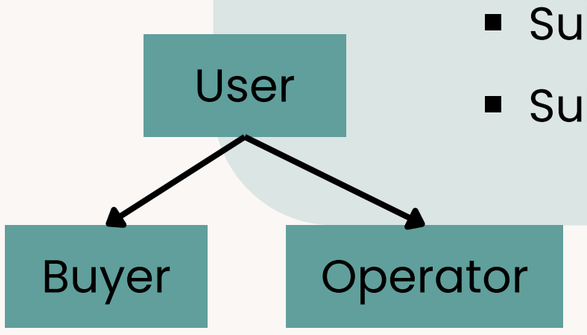
Expected Output:

- Buyer:
  - Sees item list on launch
  - Inserts cash
    → balance updates
  - Selects item → if valid:
    - Item dispensed
    - Change returned
    - Transaction logged
  - If not valid
    → error or cancel message
  - If canceled → cash returned
- Operator:
  - Restocks items
    → inventory updates
  - Updates prices
    → prices change
  - Generates report
    → saved to file

Hierarchies

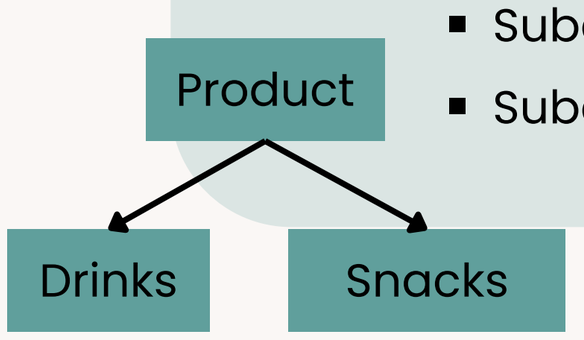
1. Product Hierarchy

- Abstract Class: User
  - Subclass: Buyer
  - Subclass: Operator



2. Product Hierarchy

- Abstract Class: Food
  - Subclass: Drink
  - Subclass: Snack



Interface:

- MessageDisplayable

  + displayMessage(String msg): void

abstracts message–display mechanism so both buyers and operators can receive prompts or errors.

- TransactionHandler

  + processTransaction(...): boolean

Handles purchase workflow, allowing future integration of online banking.

Runtime Polymorphism:

Achieved through the MessageDisplayable interface.

- Buyer, Operator, and VendingMachine each implement displayMessage() with their own specific logic.
- Ensures consistent and customizable communication across different components of the system.

TextI/O Usage:

Class: VendingMachine

Purpose: Writes daily transaction and profit summaries to a text file. Reads historical profit data from a specified file:

- writeToFile(): Writes transaction/profit summary to a text file.
- readProfitSheet(String fileName): Reads and displays historical profit data.

Class: Operator

Purpose: Reviews historical profit data stored in a file:

- reviewProfitSheet(): Displays historical profit information using readProfitSheet().

Comparable and Comparator

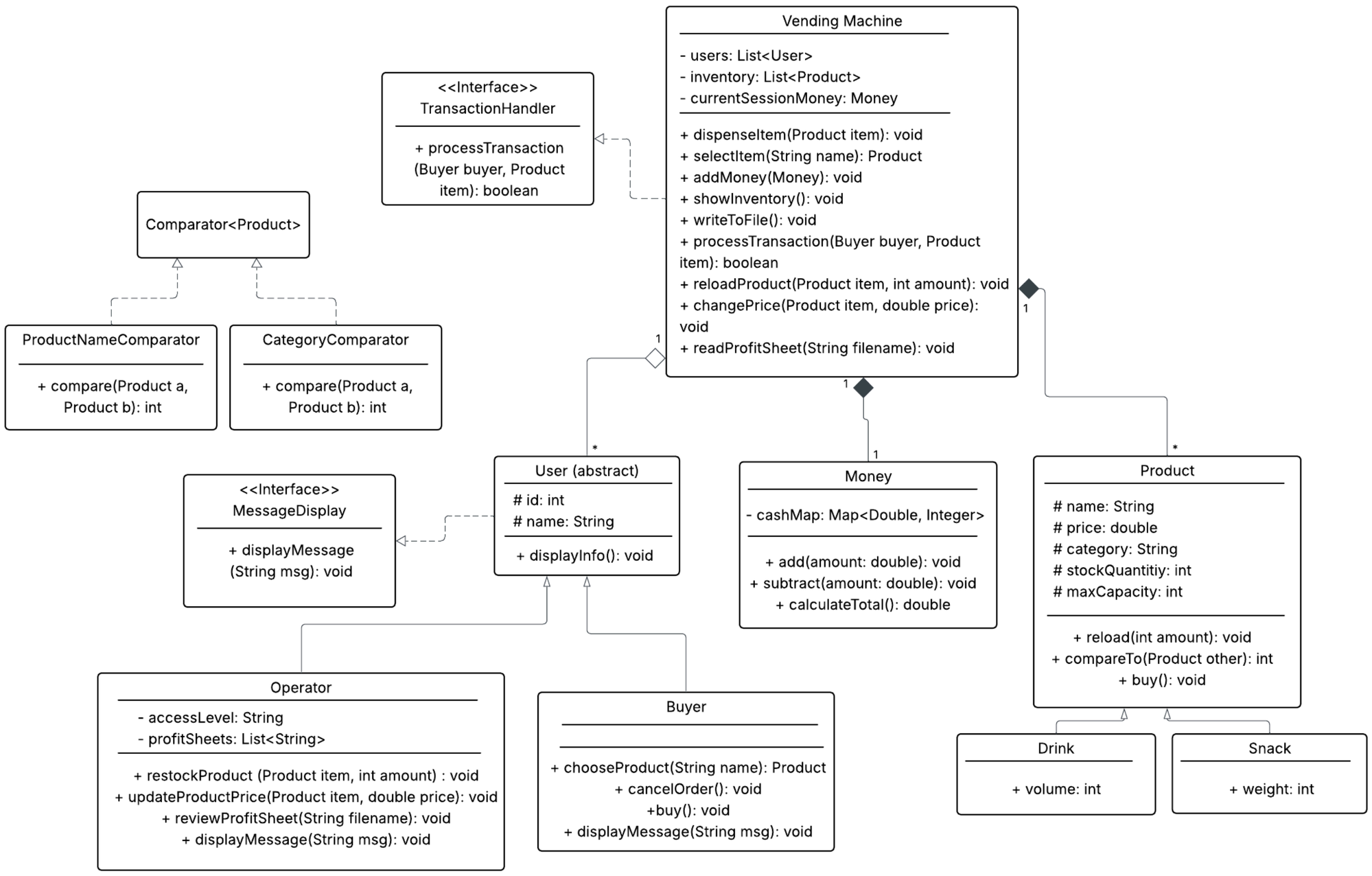
Class: Product

implements Comparable<Product> (natural ordering by price or name).

Class: ProductNameComparator and CategoryComparator

for alternative sorting strategies (e.g., alphabetically by name or grouped by category).

UML Class Diagram:



Built on Lucidchart

Deliverable 2: (50% Checkpoint)

- Class & Interface Skeletons
  - User (abstract), Buyer, Operator
  - MessageDisplayable interface
  - Product (with Comparable<Product>), Drink, Snack
  - Money, VendingMachine (implements MessageDisplayable)
  - Empty ProductNameComparator, CategoryComparator

- Implement Basic Attributes
  - Define fields (e.g., name, price, category, quantity)

- Implement Comparable & Comparator
  - compareTo() in Product(by price)
  - compare() in ProductNameComparator, CategoryComparator

- Unit Testing
  - Write unit tests for all user defined methods

- Finish writing the Money, Product class, and its subclasses